(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 March 2001 (08.03.2001)

PCT

(10) International Publication Number
WO 01/16758 A2

(51) International Patent Classification[7]: G06F 12/00

(21) International Application Number: PCT/US00/23982

(22) International Filing Date: 31 August 2000 (31.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/151,961 1 September 1999 (01.09.1999) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 60/151,961 (CIP)
Filed on 1 September 1999 (01.09.1999)

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **WOLRICH, Gilbert** [US/US]; 4 Cider Mill Road, Framingham, MA 01701 (US). **ADILETTA, Matthew, J.** [US/US]; 20 Monticello Drive, Worcester, MA 01603 (US). **WHEELER, William** [US/US]; 745 School Street, Webster, MA 01570 (US). **BERNSTEIN, Debra** [US/US]; 38 Helen Street, Waltham, MA 02452 (US). **HOOPER, Donald** [US/US]; 19 Main Circle, Shrewsbury, MA 01545 (US).

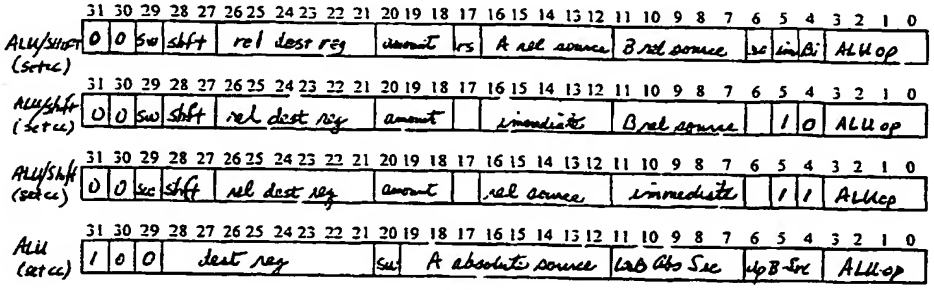
(74) Agents: **MALONEY, Denis, G.**; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 et al. (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: DOUBLE SHIFT INSTRUCTION FOR MICRO ENGINE USED IN MULTITHREADED PARALLEL PROCESSOR ARCHITECTURE



Shift Decode:
(rs, r0) decode ([31:0] shifts into [63:32] and take [63:32]):

00 = left rotate
01 = right shift (32-ShfAmt = Right Shft Amt)
10 = left shift
11 = double shift (upper A-op shifts into lower B-op)

===> "left rotate" of zero gives zero shift (therwise zero amount signifies indirect shift)

ALU-OP decode:

0000 = B	0100 = ~A&B (~and)	1000 = A-B	1100 = A+B(8)
0001 = ~B	0101 = XOR	1001 = B-A	1101 = A+B(16)
0010 = A&B (and)	0110 = OR	1010 =	1110 = A+B
0011 = A&~B (and~)	0111 = mul-stuff	1011 =	1111 = A+B+Cin

(57) Abstract: A method of operating a processor comprising concatenating a first word and a second word to produce an intermediate result, shifting the intermediate result by a specified shift amount and storing the shifted intermediate result in a third word.

WO 01/16758 A2

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DOUBLE SHIFT INSTRUCTION FOR MICRO ENGINE USED IN MULTITHREADED PARALLEL PROCESSOR ARCHITECTURE

TECHNICAL FIELD

This invention relates to a memory instruction for computer processors.

BACKGROUND

Parallel processing is an efficient form of information processing of concurrent events in a computing process. Parallel processing demands concurrent execution of many programs in a computer. in contrast to sequential processing. In the context of a parallel processor. parallelism involves doing more than one thing at the same time. Unlike a serial paradigm where all tasks are performed sequentially at a single station or a pipelined machine where tasks are performed at specialized stations. with parallel processing, a number of stations are provided with each capable of performing all tasks. That is. in general all or a number of the stations work simultaneously and independently on the same or common elements of a problem. Certain problems are suitable for solution by applying parallel processing.

DESCRIPTION OF DRAWINGS

The foregoing features and other aspects of the invention will be described further in detail by the accompanying drawings. in which:

FIG. 1 is a block diagram of a communication system employing a hardware-based multithreaded processor.

FIG. 2 is a detailed block diagram of the hardware-based multithreaded processor of FIG. 1.

FIG. 3 is a block diagram of a micro engine functional unit employed in the hardware-based multithreaded processor of FIGS. 1 and 2.

FIG. 4 is a block diagram of a pipeline in the micro engine of FIG. 3.

FIG. 5 is a block diagram illustrating a format for arithmetic logic unit instruction results.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

Referring to FIG. 1, a communication system 10 includes a parallel, hardware-based multithreaded processor 12. The hardware-based multithreaded processor 12 is coupled to a bus such as a PCI bus 14, a memory system 16 and a second bus 18. The system 10 is especially useful for tasks that can be broken into parallel subtasks or functions. Specifically, hardware-based multithreaded processor 12 is useful for tasks that are bandwidth oriented rather than latency oriented. The hardware-based multithreaded processor 12 has multiple micro engines 22 each with multiple hardware controlled threads that can be simultaneously active and independently work on a task.

The hardware-based multithreaded processor 12 also includes a central controller 20 that assists in loading micro code control for other resources of the hardware-based multithreaded processor 12 and performs other general purpose computer type functions such as handling protocols, exceptions, extra support for packet processing where the micro engines 22 pass the packets off for more detailed processing such as in boundary conditions. In one embodiment, the processor 20 is a Strong Arm® (Arm is a trademark of ARM Limited, United Kingdom) based architecture. The general-purpose microprocessor 20 has an operating system. Through the operating system the processor 20 can call functions to operate on micro engines 22a-22f. The processor 20 can use any supported operating system, preferably a real time operating system. For the core processor 20 implemented as Strong Arm architecture, operating systems such as, Microsoft-NT real-time, VXWorks and μCUS, a freeware operating system available over the Internet, can be used.

Functional micro engines (micro engines) 22a-22f each maintain program counters in hardware and states associated with the program counters. Effectively, a corresponding number of sets of threads can be simultaneously active on each of the micro engines 22a-22f while only one is actually operating at any one time.

In an embodiment, there are six micro engines 22a-22f as shown. Each micro engine 22a-22f has capabilities for processing four hardware threads. The six micro engines 22a-22f operate with shared resources including memory system 16 and bus interfaces 24 and 28. The memory system 16 includes a Synchronous Dynamic Random Access Memory (SDRAM) controller 26a and a Static Random Access Memory (SRAM) controller 26b. SDRAM memory 16a and SDRAM controller 26a are typically used for processing large volumes of data, e.g., processing of network payloads from network

packets. The SRAM controller 26b and SRAM memory 16b are used in a networking implementation for low latency, fast access tasks, e.g., accessing look-up tables, memory for the core processor 20, and so forth.

The six micro engines 22a-22f access either the SDRAM 16a or SRAM 16b based on characteristics of the data. Thus, low latency, low bandwidth data is stored in and fetched from SRAM 16b, whereas higher bandwidth data for which latency is not as important, is stored in and fetched from SDRAM 16a. The micro engines 22a-22f can execute memory reference instructions to either the SDRAM controller 26a or SRAM controller 16b.

Advantages of hardware multithreading can be explained by SRAM or SDRAM memory accesses. As an example, an SRAM access requested by a Thread_0, from a micro engine will cause the SRAM controller 26b to initiate an access to the SRAM memory 16b. The SRAM controller 26b controls arbitration for the SRAM bus, accesses the SRAM 16b, fetches the data from the SRAM 16b, and returns data to a requesting micro engine 22a-22f. During an SRAM access, if the micro engine, e.g., micro engine 22a, had only a single thread that could operate, that micro engine would be dormant until data was returned from the SRAM 16b. By employing hardware context swapping within each of the micro engines 22a-22f, the hardware context swapping enables other contexts with unique program counters to execute in that same micro engine. Thus, another thread, e.g., Thread_1 can function while the first thread, i.e., Thread_0, is awaiting the read data to return. During execution, Thread_1 may access the SDRAM memory 16a. While Thread_1 operates on the SDRAM unit 16a, and Thread_0 is operating on the SRAM unit 16b, a new thread, e.g., Thread_2 can now operate in the micro engine 22a. Thread_2 can operate for a certain amount of time until it needs to access memory or perform some other long latency operation, such as making an access to a bus interface. Therefore, simultaneously, the processor 12 can have a bus operation, SRAM operation and SDRAM operation all being completed or operated upon by one micro engine 22a and have one more thread available to process more work in the data path.

The hardware context swapping also synchronizes completion of tasks. For example, two threads could hit the same shared resource e.g., SRAM 16b. Each one of these separate functional units, e.g., the FBUS interface 28, the SRAM controller 26a, and the SDRAM controller 26b, when they complete a requested task from one of the micro engine thread contexts reports back a flag signaling completion of an operation. When

the micro engine receives the flag, the micro engine can determine which thread to turn on.

An application for the hardware-based multithreaded processor 12 is as a network processor. As a network processor, the hardware-based multithreaded processor 12 interfaces to network devices such as a media access controller device e.g., a 10/100BaseT Octal MAC 13a or a Gigabit Ethernet device 13b. In general, as a network processor, the hardware-based multithreaded processor 12 can interface to any type of communication device or interface that receives/sends large amounts of data. Communication system 10 functioning in a networking application could receive network packets from the devices 13a, 13b and process those packets in a parallel manner. With the hardware-based multithreaded processor 12, each network packet can be independently processed.

Another example for use of processor 12 is a print engine for a postscript processor or as a processor for a storage subsystem, e.g., Redundant Array of Independent Disk (RAID) storage, a category of disk drives that employs two or more drives in combination for fault tolerance and performance. A further use is as a matching engine. In the securities industry for example, the advent of electronic trading requires the use of electronic matching engines to match orders between buyers and sellers. These and other parallel types of tasks can be accomplished utilizing the system 10.

The processor 12 includes the bus interface 28 that couples the processor to the second bus 18. In an embodiment, bus interface 28 couples the processor 12 to the FBUS (FIFO bus) 18. The FBUS interface 28 is responsible for controlling and interfacing the processor 12 to the FBUS 18. The FBUS 18 is a 64-bit wide FIFO bus, used to interface to Media Access Controller (MAC) devices, e.g., 10/100 Base T Octal MAC 13a.

The processor 12 includes a second interface e.g., PCI bus interface 24, that couples other system components that reside on the PCI 14 bus to the processor 12. The PCI bus interface 24 provides a high-speed data path 24a to memory 16, e.g., SDRAM memory 16a. Through PCI bus interface 24 data can be moved quickly from the SDRAM 16a through the PCI bus 14, via direct memory access (DMA) transfers. The hardware based multithreaded processor 12 supports image transfers. The hardware based multithreaded processor 12 can employ DMA channels so if one target of a DMA transfer is busy, another one of the DMA channels can take over the PCI bus 14 to deliver information to another target to maintain high processor 12 efficiency. Additionally, the

PCI bus interface 24 supports target and master operations. Target operations are operations where slave devices on bus 14 access SDRAMs through reads and writes that are serviced as a slave to a target operation. In master operations, the processor core 20 sends data directly to or receives data directly from the PCI interface 24.

Each of the functional units 22 is coupled to one or more internal buses. As described below, the internal buses are dual, 32 bit buses (i.e., one bus for read and one for write). The hardware-based multithreaded processor 12 also is constructed such that the sum of the bandwidths of the internal buses in the processor 12 exceed the bandwidth of external buses coupled to the processor 12. The processor 12 includes an internal core processor bus 32, e.g., an ASB Advanced System Bus (ASB), that couples the processor core 20 to the memory controller 26a, 26b and to an ASB translator 30, described below. The ASB bus 32 is a subset of the so-called Advanced Microcontroller Bus Architecture (AMBA) bus that is used with the Strong Arm processor core 20. AMBA is an open standard, on-chip bus specification that details a strategy for the interconnection and management of functional blocks that makes up a System-on-chip (SoC). The processor 12 also includes a private bus 34 that couples the micro engine units 22 to SRAM controller 26b, ASB translator 30 and FBUS interface 28. A memory bus 38 couples the memory controller 26a, 26b to the bus interfaces 24 and 28 and memory system 16 including flashrom 16c that is used for boot operations and so forth.

Referring to FIG. 2, each of the micro engines 22a-22f includes an arbiter that examines flags to determine the available threads to be operated upon. Any thread from any of the micro engines 22a-22f can access the SDRAM controller 26a, SDRAM controller 26b or FBUS interface 28. The memory controllers 26a and 26b each include queues to store outstanding memory reference requests. The queues either maintain order of memory references or arrange memory references to optimize memory bandwidth. For example, if a thread_0 has no dependencies or relationship to a thread_1, there is no reason that thread_1 and thread_0 cannot complete their memory references to the SRAM unit 16b out of order. The micro engines 22a-22f issue memory reference requests to the memory controllers 26a and 26b. The micro engines 22a-22f flood the memory subsystems 26a and 26b with enough memory reference operations such that the memory subsystems 26a and 26b become the bottleneck for processor 12 operation.

If the memory subsystem 16 is flooded with memory requests that are independent in nature, the processor 12 can perform memory reference sorting. Memory reference

sorting improves achievable memory bandwidth. Memory reference sorting, as described below, reduces dead time or a bubble that occurs with accesses to SRAM 16b. With memory references to SRAM 16b, switching current direction on signal lines between reads and writes produces a bubble or a dead time waiting for current to settle on conductors coupling the SRAM 16b to the SRAM controller 26b.

That is, the drivers that drive current on the bus need to settle out prior to changing states. Thus, repetitive cycles of a read followed by a write can degrade peak bandwidth. Memory reference sorting allows the processor 12 to organize references to memory such that long strings of reads can be followed by long strings of writes. This can be used to minimize dead time in the pipeline to effectively achieve closer to maximum available bandwidth. Reference sorting helps maintain parallel hardware context threads. On the SDRAM 16a, reference sorting allows hiding of pre-charges from one bank to another bank. Specifically, if the memory system 16b is organized into an odd bank and an even bank, while the processor is operating on the odd bank, the memory controller can start pre-charging the even bank. Pre-charging is possible if memory references alternate between odd and even banks. By ordering memory references to alternate accesses to opposite banks, the processor 12 improves SDRAM bandwidth. Additionally, other optimizations can be used. For example, merging optimizations where operations that can be merged, are merged prior to memory access, open page optimizations where by examining addresses an opened page of memory is not reopened, chaining, as will be described below, and refreshing mechanisms, can be employed.

The FBUS interface 28 supports Transmit and Receive flags for each port that a MAC device supports, along with an Interrupt flag indicating when service is warranted. The FBUS interface 28 also includes a controller 28a that performs header processing of incoming packets from the FBUS 18. The controller 28a extracts the packet headers and performs a micro programmable source/destination/protocol hashed lookup (used for address smoothing) in SRAM 16b. If the hash does not successfully resolve, the packet header is sent to the processor core 20 for additional processing. The FBUS interface 28 supports the following internal data transactions:

FBUS unit	(Shared bus SRAM)	to/from micro engine.
FBUS unit	(via private bus)	writes from SDRAM Unit.
FBUS unit	(via Mbus)	Reads to SDRAM.

The FBUS 18 is a standard industry bus and includes a data bus, e.g., 64 bits wide and sideband control for address and read/write control. The FBUS interface 28 provides the ability to input large amounts of data using a series of input and output FIFOs 29a-29b. From the FIFOs 29a-29b, the micro engines 22a-22f fetch data from or command the SDRAM controller 26a to move data from a receive FIFO in which data has come from a device on bus 18, into the FBUS interface 28. The data can be sent through memory controller 26a to SDRAM memory 16a, via a direct memory access. Similarly, the micro engines can move data from the SDRAM 26a to interface 28, out to FBUS 18, via the FBUS interface 28.

Data functions are distributed amongst the micro engines 22. Connectivity to the SRAM 26a, SDRAM 26b and FBUS 28 is via command requests. A command request can be a memory request or a FBUS request. For example, a command request can move data from a register located in a micro engine 22a to a shared resource, e.g., an SDRAM location, SRAM location, flash memory or some MAC address. The commands are sent out to each of the functional units and the shared resources. However, the shared resources do not need to maintain local buffering of the data. Rather, the shared resources access distributed data located inside of the micro engines 22a-22f. This enables micro engines 22a-22f, to have local access to data rather than arbitrating for access on a bus and risk contention for the bus. With this feature, there is a zero cycle stall for waiting for data internal to the micro engines 22a-22f.

The data buses, e.g., ASB bus 30, SRAM bus 34 and SDRAM bus 38 coupling these shared resources, e.g., memory controllers 26a and 26b, are of sufficient bandwidth such that there are no internal bottlenecks. In order to avoid bottlenecks, the processor 12 has a bandwidth requirement where each of the functional units is provided with at least twice the maximum bandwidth of the internal buses. As an example, the SDRAM 16a can run a 64 bit wide bus at 83 MHz. The SRAM data bus could have separate read and write buses, e.g., could be a read bus of 32 bits wide running at 166 MHz and a write bus of 32 bits wide at 166 MHz. That is, in essence, 64 bits running at 166 MHz that is effectively twice the bandwidth of the SDRAM.

The core processor 20 also can access the shared resources. The core processor 20 has a direct communication to the SDRAM controller 26a to the bus interface 24 and to SRAM controller 26b via bus 32. However, to access the micro engines 22a-22f and transfer registers located at any of the micro engines 22a-22f, the core processor 20 access the micro engines 22a-22f via the ASB Translator 30 over bus 34. The ASB translator 30 can physically reside in the FBUS interface 28, but logically is distinct. The ASB Translator 30 performs an address translation between FBUS micro engine transfer register locations and core processor addresses (i.e., ASB bus) so that the core processor 20 can access registers belonging to the micro engines 22a-22f.

Although micro engines 22a-22f can use the register set to exchange data as described below, a scratchpad memory 27 is also provided to permit micro engines 22a-22f to write data out to the memory for other micro engines to read. The scratchpad 27 is coupled to bus 34.

The processor core 20 includes a RISC core 50 implemented in a five stage pipeline performing a single cycle shift of one operand or two operands in a single cycle, provides multiplication support and 32 bit barrel shift support. This RISC core 50 is a standard Strong Arm® architecture but it is implemented with a five-stage pipeline for performance reasons. The processor core 20 also includes a 16-kilobyte instruction cache 52, an 8-kilobyte data cache 54 and a prefetch stream buffer 56. The core processor 20 performs arithmetic operations in parallel with memory writes and instruction fetches. The core processor 20 interfaces with other functional units via the ARM defined ASB bus. The ASB bus is a 32-bit bi-directional bus 32.

Referring to FIG. 3, an exemplary one of the micro engines 22a-22f, e.g., micro engine 22f, is shown. The micro engine 22f includes a control store 70, which, in one implementation, includes a RAM of here 1,024 words of 32 bit. The RAM stores a micro program (not shown). The micro program is loadable by the core processor 20. The micro engine 22f also includes controller logic 72. The controller logic 72 includes an instruction decoder 73 and program counter (PC) units 72a-72d. The four micro program counters 72a-72d are maintained in hardware. The micro engine 22f also includes context event switching logic 74. Context event logic 74 receives messages (e.g., SEQ_#_EVENT_RESPONSE; FBI_EVENT_RESPONSE; SRAM _EVENT_RESPONSE; SDRAM _EVENT_RESPONSE; and ASB _EVENT_RESPONSE) from each one of the shared resources, e.g., SRAM 26a, SDRAM

26b, or processor core 20, control and status registers, and so forth. These messages provide information on whether a requested function has completed. Based on whether or not a function requested by a thread has completed and signaled completion, the thread needs to wait for that completion signal, and if the thread is enabled to operate, then the thread is placed on an available thread list (not shown). The micro engine 22f can have a maximum of four threads available.

In addition to event signals that are local to an executing thread, the micro engines 22a-22f employ signaling states that are global. With signaling states, an executing thread can broadcast a signal state to all micro engines 22a-22f, e.g., Receive Request Available (RRA) signal, any and all threads in the micro engines 22a-22f can branch on these signaling states. These signaling states can be used to determine availability of a resource or whether a resource is due for servicing.

The context event logic 74 has arbitration for the four threads. In an embodiment, the arbitration is a round robin mechanism. Other techniques could be used including priority queuing or weighted fair queuing. The micro engine 22f also includes an execution box (EBOX) data path 76 that includes an arithmetic logic unit (ALU) 76a and general-purpose register set 76b. The ALU 76a performs arithmetic and logical functions as well as shift functions. The register set 76b has a relatively large number of general-purpose registers. In an embodiment, there are 64 general-purpose registers in a first bank, Bank A and 64 in a second bank, Bank B. The general-purpose registers are windowed so that they are relatively and absolutely addressable.

The micro engine 22f also includes a write transfer register stack 78 and a read transfer stack 80. These registers 78 and 80 are also windowed so that they are relatively and absolutely addressable. Write transfer register stack 78 is where write data to a resource is located. Similarly, read register stack 80 is for return data from a shared resource. Subsequent to or concurrent with data arrival, an event signal from the respective shared resource e.g., the SRAM controller 26a, SDRAM controller 26b or core processor 20 will be provided to context event arbiter 74, which will then alert the thread that the data is available or has been sent. Both transfer register banks 78 and 80 are connected to the execution box (EBOX) 76 through a data path. In an embodiment, the read transfer register has 64 registers and the write transfer register has 64 registers.

Referring to FIG. 4, the micro engine data path maintains a 5-stage micro-pipeline 82. This pipeline includes lookup of microinstruction words 82a, formation of the

register file addresses 82b, read of operands from register file 82c, ALU shift or compare operations 82d, and write-back of results to registers 82e. By providing a write-back data bypass into the ALU/shifter units, and by assuming the registers are implemented as a register file (rather than a RAM), the micro engine 22f can perform a simultaneous register file read and write, which completely hides the write operation.

The SDRAM interface 26a provides a signal back to the requesting micro engine on reads that indicates whether a parity error occurred on the read request. The micro engine micro code is responsible for checking the SDRAM 16a read Parity flag when the micro engine uses any return data. Upon checking the flag, if it was set, the act of branching on it clears it. The Parity flag is only sent when the SDRAM 16a is enabled for checking, and the SDRAM 16a is parity protected. The micro engines 22 and the PCI Unit 14 are the only requestors notified of parity errors. Therefore, if the processor core 20 or FIFO 18 requires parity protection, a micro engine assists in the request. The micro engines 22a-22f support conditional branches.

Referring to FIG. 5, a format for arithmetic logic unit instruction is shown. The micro engines 22 support various instruction sets. The instruction set includes logical and arithmetic operations that perform an ALU operation on one or two operands and deposit the result into the destination register, and update all ALU condition codes according to the result of the operation. Condition codes are lost during context swaps. When the op code bits 28:27 are 1:1 the instruction is a double shift instruction.

The instruction set includes a double shift instruction, i.e., DBL_SHF, which concatenates two long words (i.e., two 32 bit words) and shifts the result and saves the result as a longword. In the double shift instruction, the upper A-op shifts into lower B-op, with a "left rotate" of zero giving a zero shift (otherwise zero amount signifies indirect shift). The DBL_SHF instruction loads a destination register with a 32-bit longword that is formed by concatenating the A operands and B operands together, right shifting the 64-bit quantity by the specified amount, and storing the lower 32 bits.

A format of the double shift instruction is: *dbl_shf[dest_reg, A_operand, B_operand, A_op_shf_cntl]*, where each of the fields is described fully below.

A "dest_req" field represents the destination, i.e., an absolute or context-relative register name.

A "A_operand" field represents a context-relative register name, i.e., 5-bit zero-filled immediate data.

A "B_oprand" field represents a context-relative register name, i.e., 5-bit zero-filled immediate data.

A "A_op_shf_cntl" field represents a right shift of values from 1 to 31.

By way of example, if a = 0x87654321 and b = 0xFEDCBA98, then dbl_shf[c, a, b, >>12] stores 0x321FEDCB in c. The ALU condition codes are updated based on the result.

It is to be understood that while the invention has been described in conjunction with the detailed description thereof, the foregoing description is intended to illustrate and not limit the scope of the invention, which is defined by the scope of the appended claims. Other aspects, advantages, and modifications are within the scope of the following claims.

What is claimed is:

1. A method of operating a processor comprising:
concatenating a first word and a second word to produce an intermediate result;
shifting the intermediate result by a specified shift amount; and
storing the shifted intermediate result in a third word.

2. The method of claim 1 wherein the first word and the second word and the third word are 32-bit words.

3. The method of claim 1 wherein the intermediate result is a 64-bit word.

4. The method of claim 1 wherein shifting comprises right shifting.

5. The method of claim 4 wherein the specified shift amount is in an operand.

6. The method of claim 4 wherein the specified shift amount is a value between one and thirty-one.

7. The method of claim 4 wherein the specified shift amount is a value contained in a lower five bits of the first word.

8. A computer instruction comprising:
an instruction to concatenate a first word and a second word to produce an intermediate result;
shift the intermediate result by a specified amount; and
store the shifted intermediate result in a third word.

9. The instruction of claim 8 wherein the first word and the second word and the third word are 32-bit words.

10. The instruction of claim 8 wherein the intermediate result is a 64-bit word.

11. The instruction of claim 8 wherein shifting comprises right shifting.

12. The instruction of claim 11 wherein the specified amount is in an operand.

13. The instruction of claim 11 wherein the specified amount is a value between one and thirty-one.

14. The instruction of claim 11 wherein the specified amount is a value contained in a lower five bits of the first word.

5

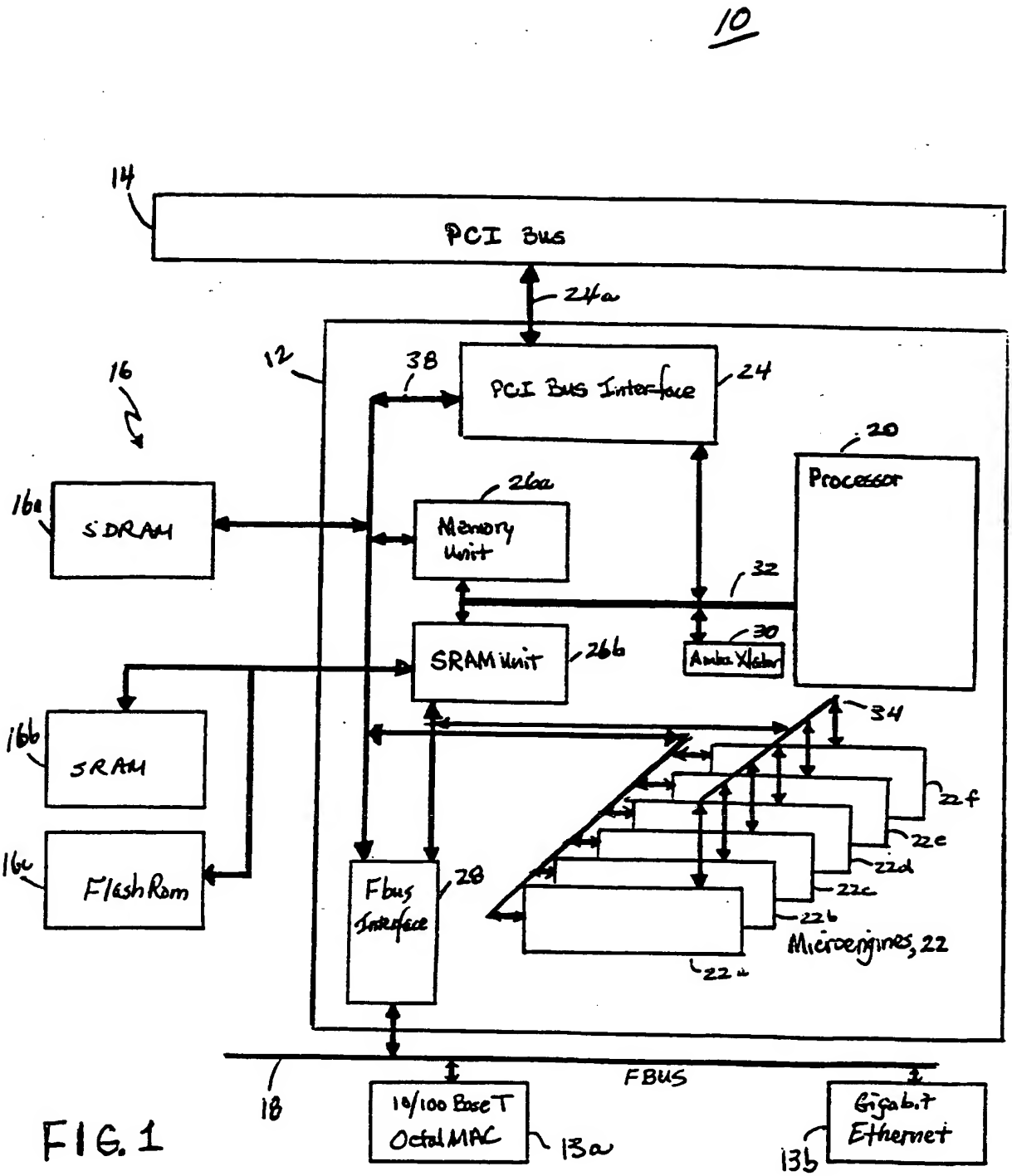

10
14
PCI Bus
24a
12
38
PCI Bus Interface
24
16
20
Processor
16a
SDRAM
26a
Memory Unit
32
SRAM Unit
26b
30
Arbiter
16b
SRAM
34
16c
Flash Rom
Fbus Interface
28
22f
22e
22d
22c
22b
22a
Microengines, 22
FBUS
18
10/100 BaseT Octal MAC
13a
Gigabit Ethernet
13b

FIG. 1

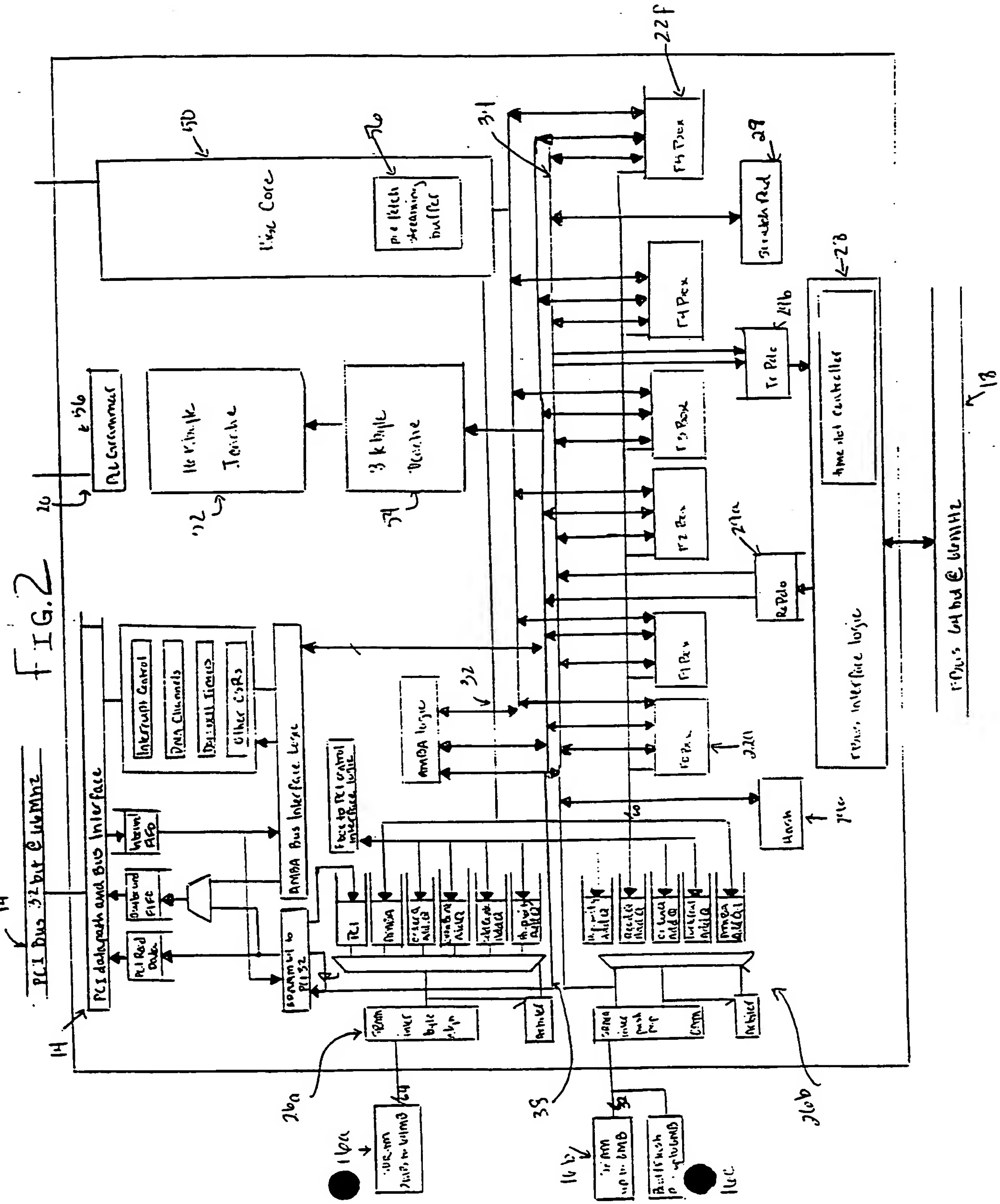

PCI Bus 32 bit @ 66MHz
PCI datapath and Bus Interface
AMBA Bus Interface Logic
AMBA logic
Ethernet controller
FBus interface logic
FBus 64 bit @ 66MHz
Hash

FIG. 2

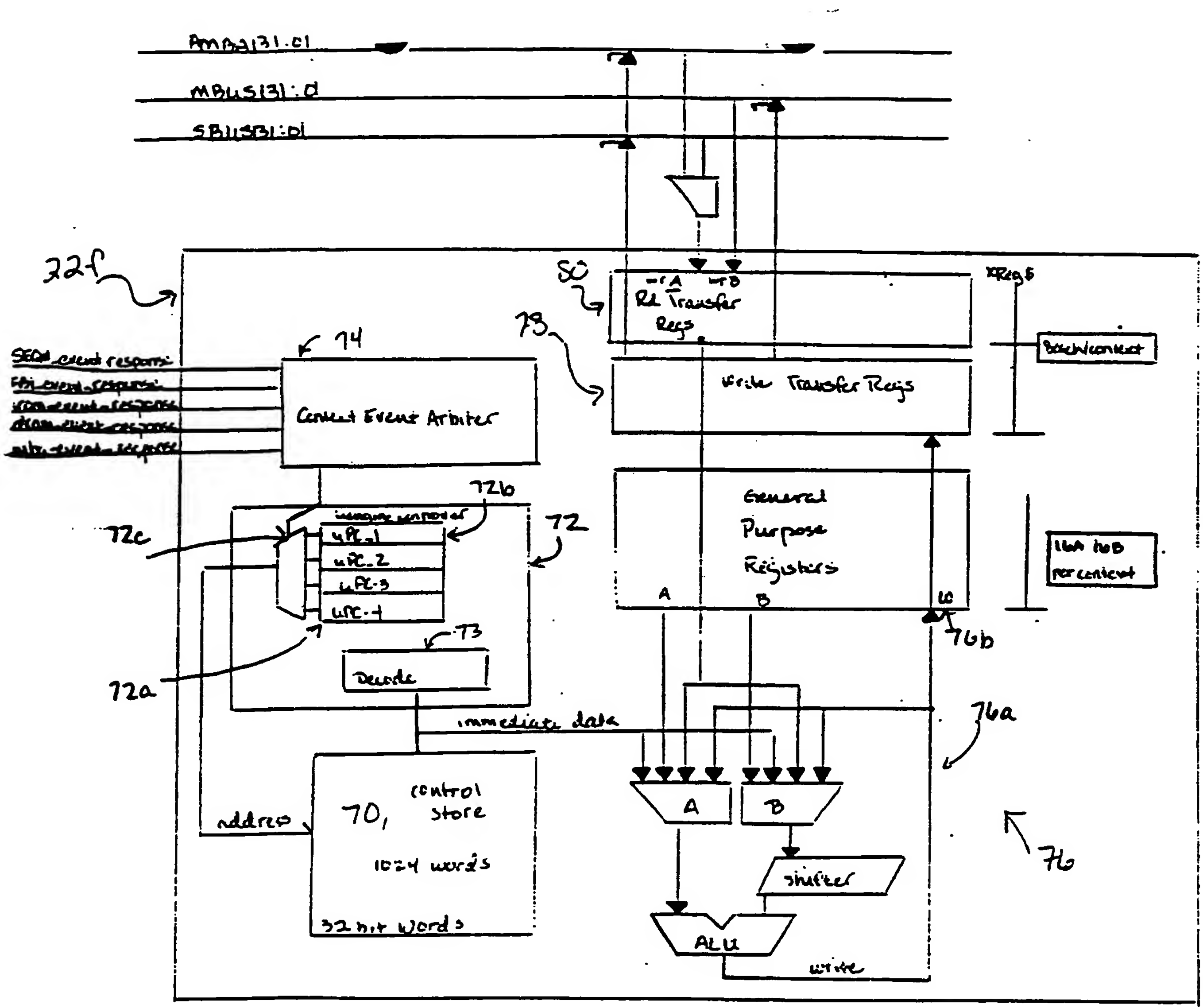

MBUS[31:0]
SBUS[31:0]
22f
80
Rd Transfer Regs
XRegs
78
Write Transfer Regs
Bank/context
74
Context Event Arbiter
72b
72c
72
uPC-1
uPC-2
uPC-3
uPC-4
73
Decode
72a
General Purpose Registers
A
B
per context
76b
76a
immediate data
address
70
control store
1024 words
32 bit words
A
B
shifter
ALU
write
76

Fig 3

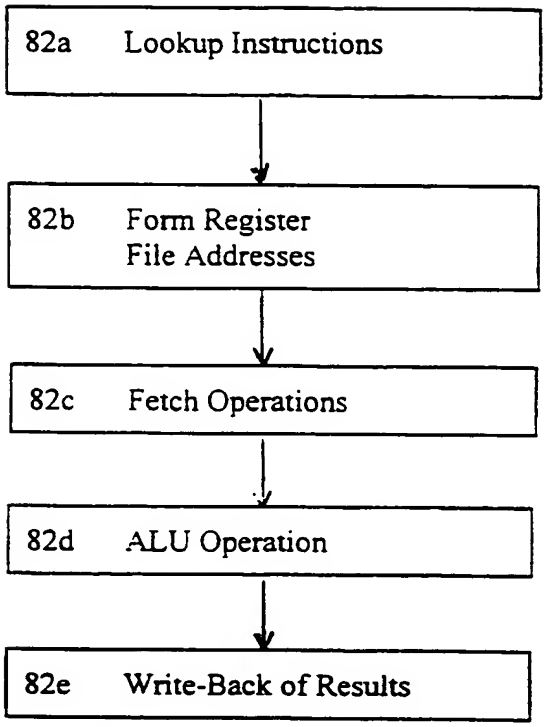
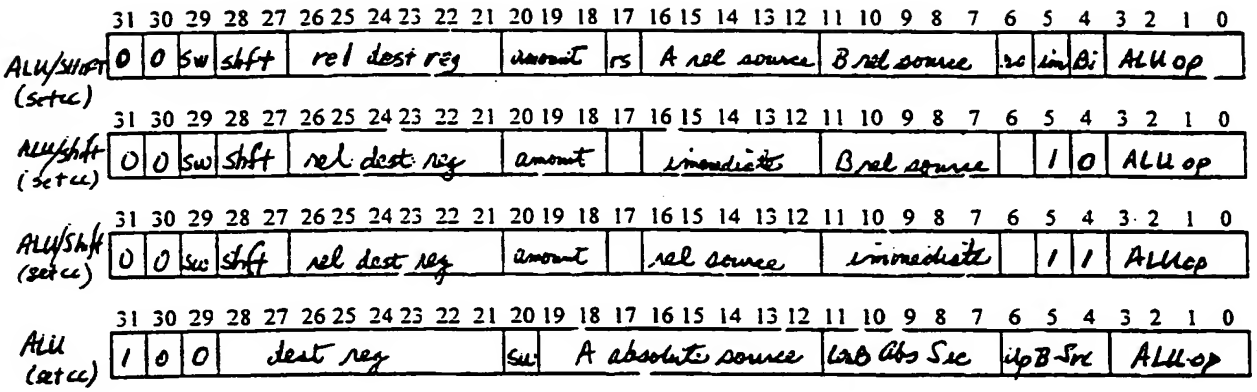

82a Lookup Instructions
82b Form Register File Addresses
82c Fetch Operations
82d ALU Operation
82e Write-Back of Results

FIG. 4



Shift Decode:

(rs, r0) decode ([31:0] shifts into [63:32] and take [63:32]):

00 = left rotate
01 = right shift (32-ShfAmt = Right Shft Amt)
10 = left shift
11 = double shift (upper A-op shifts into lower B-op)

===> "left rotate" of zero gives zero shift (therwise zero amount signifies indirect shift)

ALU-OP decode:

0000 = B	0100 = ~A&B (~and)	1000 = A-B	1100 = A+B(8)
0001 = ~B	0101 = XOR	1001 = B-A	1101 = A+B(16)
0010 = A&B (and)	0110 = OR	1010 =	1110 = A+B
0011 = A&~B (and~)	0111 = mul-stuff	1011 =	1111 = A+B+Cin

FIG. 5

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 March 2001 (08.03.2001)

PCT

(10) International Publication Number
WO 01/16758 A3

(51) International Patent Classification[7]: G06F 9/315

(21) International Application Number: PCT/US00/23982

(22) International Filing Date: 31 August 2000 (31.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/151,961 1 September 1999 (01.09.1999) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 60/151,961 (CIP)
Filed on 1 September 1999 (01.09.1999)

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **WOLRICH, Gilbert** [US/US]; 4 Cider Mill Road, Framingham, MA 01701 (US). **ADILETTA, Matthew, J.** [US/US]; 20 Monticello Drive, Worcester, MA 01603 (US). **WHEELER, William** [US/US]; 745 School Street, Webster, MA 01570 (US). **BERNSTEIN, Debra** [US/US]; 38 Helen Street, Waltham, MA 02452 (US). **HOOPER, Donald** [US/US]; 19 Main Circle, Shrewsbury, MA 01545 (US).

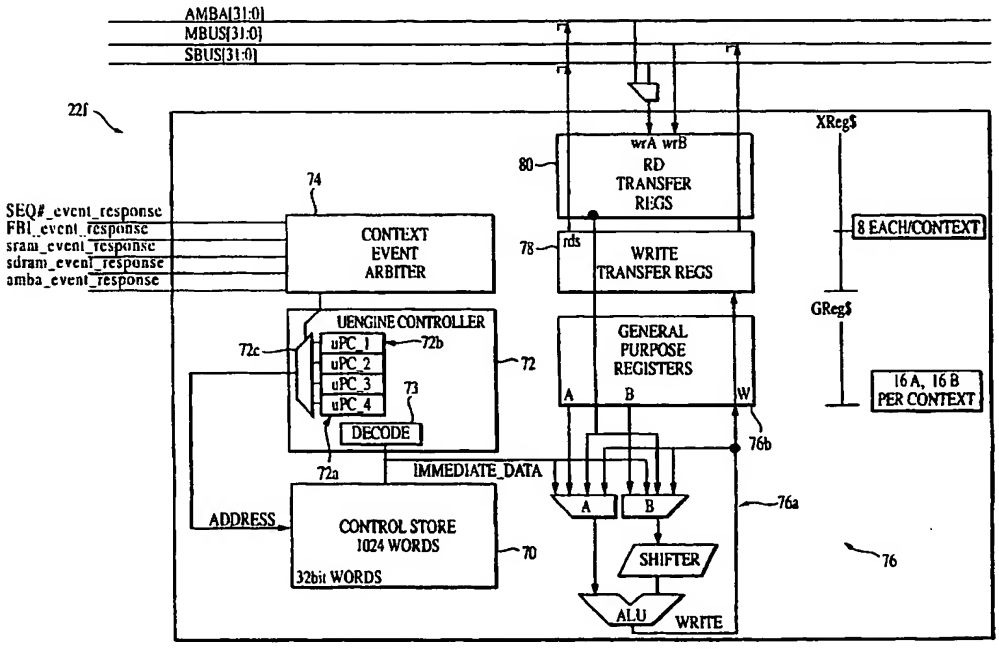
(74) Agents: **MALONEY, Denis, G.**; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 et al. (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: DOUBLE SHIFT INSTRUCTION FOR MICRO ENGINE USED IN MULTITHREADED PARALLEL PROCESSOR ARCHITECTURE



(57) Abstract: A method of operating a processor comprises concatenating a first word and a second word to produce an intermediate result. An ALU (76a) shifts the intermediate result by a specified shift amount and stores, in a register (76b), the shifted intermediate result in a third word.

WO 01/16758 A3

Published:
— *with international search report*

(88) Date of publication of the international search report:
25 October 2001

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

Internat[illegible]pplication No.
PCT/US00/23982

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 9/315
US CL :712/204, 24; 711/220, 219

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 712/204, 24; 711/220, 219

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,717,760 A (SATTERFIELD et al.) 10 February 1998, note e.g. Figure 16.	1-14
Y	US 5,652,583 A (KANG) 29 July 1997, Figs 1 and 3, the Abstract and col.2.	1-14
Y	US 5,600,812 A (PARK) 04 February 1997, note the Abstract and cols. 3-5.	1-14
Y	US 5,436,626 A (FUJIWARA et al.) 25 July 1995, note the Abstract and Figure 2.	1-14
Y	US 5,363,448 A (KOOPMAN, Jr. et al.) 08 November 1994, note the Abstract and cols. 3-4.	1-14
A	US 5,113,516 A (JOHNSON) 12 May 1992, note the Abstract.	1-14

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
14 DECEMBER 2000	12 APR 2001

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Peggy Harwood B. JAMES PEIKARI
Facsimile No. (703) 305-3230	Telephone No. (703) 305-3824

INTERNATIONAL SEARCH REPORT

Intematic[redacted]pplication No.
PCT/US00/23982

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 4,023,023 A (BOURREZ et al.) 10 May 1977, note the Abstract.	1-14

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

CORRECTED VERSION

10070006

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 March 2001 (08.03.2001)

PCT

(10) International Publication Number
WO 01/016758 A3

(51) International Patent Classification[7]: G06F 9/315

(21) International Application Number: PCT/US00/23982

(22) International Filing Date: 31 August 2000 (31.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/151,961 1 September 1999 (01.09.1999) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 60/151,961 (CIP)
Filed on 1 September 1999 (01.09.1999)

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors; and

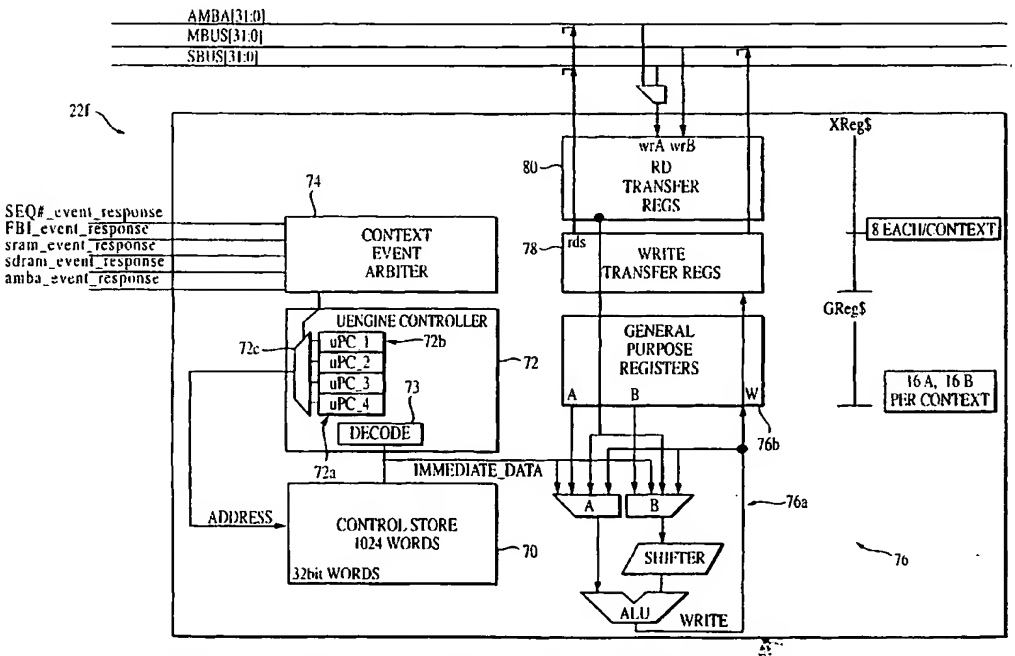
(75) Inventors/Applicants *(for US only)*: **WOLRICH, Gilbert** [US/US]; 4 Cider Mill Road, Framingham, MA 01701 (US). **ADILETTA, Matthew, J.** [US/US]; 20 Monticello Drive, Worcester, MA 01603 (US). **WHEELER, William** [US/US]; 745 School Street, Webster, MA 01570 (US). **BERNSTEIN, Debra** [US/US]; 38 Helen Street, Waltham, MA 02452 (US). **HOOPER, Donald** [US/US]; 19 Main Circle, Shrewsbury, MA 01545 (US).

(74) Agents: **MALONEY, Denis, G.**; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 et al. (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: DOUBLE SHIFT INSTRUCTION FOR MICRO ENGINE USED IN MULTITHREADED PARALLEL PROCESSOR ARCHITECTURE



(57) Abstract: A method of operating a processor comprises concatenating a first word and a second word to produce an intermediate result. An ALU (76a) shifts the intermediate result by a specified shift amount and stores, in a register (76b), the shifted intermediate result in a third word.

WO 01/016758 A3

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

(88) **Date of publication of the international search report:**
25 October 2001

(48) **Date of publication of this corrected version:**
12 September 2002

(15) **Information about Correction:**
see PCT Gazette No. 37/2002 of 12 September 2002, Section II

For two-letter codes and other abbreviations. refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DOUBLE SHIFT INSTRUCTION FOR MICRO ENGINE USED IN MULTITHREADED PARALLEL PROCESSOR ARCHITECTURE

TECHNICAL FIELD

This invention relates to a memory instruction for computer processors.

BACKGROUND

Parallel processing is an efficient form of information processing of concurrent events in a computing process. Parallel processing demands concurrent execution of many programs in a computer, in contrast to sequential processing. In the context of a parallel processor, parallelism involves doing more than one thing at the same time. Unlike a serial paradigm where all tasks are performed sequentially at a single station or a pipelined machine where tasks are performed at specialized stations, with parallel processing, a number of stations are provided with each capable of performing all tasks. That is, in general all or a number of the stations work simultaneously and independently on the same or common elements of a problem. Certain problems are suitable for solution by applying parallel processing.

DESCRIPTION OF DRAWINGS

The foregoing features and other aspects of the invention will be described further in detail by the accompanying drawings, in which:

FIG. 1 is a block diagram of a communication system employing a hardware-based multithreaded processor.

FIG. 2 is a detailed block diagram of the hardware-based multithreaded processor of FIG. 1.

FIG. 3 is a block diagram of a micro engine functional unit employed in the hardware-based multithreaded processor of FIGS. 1 and 2.

FIG. 4 is a block diagram of a pipeline in the micro engine of FIG. 3.

FIG. 5 is a block diagram illustrating a format for arithmetic logic unit instruction results.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

Referring to FIG. 1, a communication system 10 includes a parallel, hardware-based multithreaded processor 12. The hardware-based multithreaded processor 12 is coupled to a bus such as a PCI bus 14, a memory system 16 and a second bus 18. The system 10 is especially useful for tasks that can be broken into parallel subtasks or functions. Specifically, hardware-based multithreaded processor 12 is useful for tasks that are bandwidth oriented rather than latency oriented. The hardware-based multithreaded processor 12 has multiple micro engines 22 each with multiple hardware controlled threads that can be simultaneously active and independently work on a task.

The hardware-based multithreaded processor 12 also includes a central controller 20 that assists in loading micro code control for other resources of the hardware-based multithreaded processor 12 and performs other general purpose computer type functions such as handling protocols, exceptions, extra support for packet processing where the micro engines 22 pass the packets off for more detailed processing such as in boundary conditions. In one embodiment, the processor 20 is a Strong Arm® (Arm is a trademark of ARM Limited, United Kingdom) based architecture. The general-purpose microprocessor 20 has an operating system. Through the operating system the processor 20 can call functions to operate on micro engines 22a-22f. The processor 20 can use any supported operating system, preferably a real time operating system. For the core processor 20 implemented as Strong Arm architecture, operating systems such as, Microsoft-NT real-time, VXWorks and μCUS, a freeware operating system available over the Internet, can be used.

Functional micro engines (micro engines) 22a-22f each maintain program counters in hardware and states associated with the program counters. Effectively, a corresponding number of sets of threads can be simultaneously active on each of the micro engines 22a-22f while only one is actually operating at any one time.

In an embodiment, there are six micro engines 22a-22f as shown. Each micro engine 22a-22f has capabilities for processing four hardware threads. The six micro engines 22a-22f operate with shared resources including memory system 16 and bus interfaces 24 and 28. The memory system 16 includes a Synchronous Dynamic Random Access Memory (SDRAM) controller 26a and a Static Random Access Memory (SRAM) controller 26b. SDRAM memory 16a and SDRAM controller 26a are typically used for processing large volumes of data, e.g., processing of network payloads from network

packets. The SRAM controller 26b and SRAM memory 16b are used in a networking implementation for low latency, fast access tasks, e.g., accessing look-up tables, memory for the core processor 20, and so forth.

The six micro engines 22a-22f access either the SDRAM 16a or SRAM 16b based on characteristics of the data. Thus, low latency, low bandwidth data is stored in and fetched from SRAM 16b, whereas higher bandwidth data for which latency is not as important, is stored in and fetched from SDRAM 16a. The micro engines 22a-22f can execute memory reference instructions to either the SDRAM controller 26a or SRAM controller 16b.

Advantages of hardware multithreading can be explained by SRAM or SDRAM memory accesses. As an example, an SRAM access requested by a Thread_0, from a micro engine will cause the SRAM controller 26b to initiate an access to the SRAM memory 16b. The SRAM controller 26b controls arbitration for the SRAM bus, accesses the SRAM 16b, fetches the data from the SRAM 16b, and returns data to a requesting micro engine 22a-22f. During an SRAM access, if the micro engine, e.g., micro engine 22a, had only a single thread that could operate, that micro engine would be dormant until data was returned from the SRAM 16b. By employing hardware context swapping within each of the micro engines 22a-22f, the hardware context swapping enables other contexts with unique program counters to execute in that same micro engine. Thus, another thread, e.g., Thread_1 can function while the first thread, i.e., Thread_0, is awaiting the read data to return. During execution, Thread_1 may access the SDRAM memory 16a. While Thread_1 operates on the SDRAM unit 16a, and Thread_0 is operating on the SRAM unit 16b, a new thread, e.g., Thread_2 can now operate in the micro engine 22a. Thread_2 can operate for a certain amount of time until it needs to access memory or perform some other long latency operation, such as making an access to a bus interface. Therefore, simultaneously, the processor 12 can have a bus operation, SRAM operation and SDRAM operation all being completed or operated upon by one micro engine 22a and have one more thread available to process more work in the data path.

The hardware context swapping also synchronizes completion of tasks. For example, two threads could hit the same shared resource e.g., SRAM 16b. Each one of these separate functional units, e.g., the FBUS interface 28, the SRAM controller 26a, and the SDRAM controller 26b, when they complete a requested task from one of the micro engine thread contexts reports back a flag signaling completion of an operation. When

the micro engine receives the flag, the micro engine can determine which thread to turn on.

An application for the hardware-based multithreaded processor 12 is as a network processor. As a network processor, the hardware-based multithreaded processor 12 interfaces to network devices such as a media access controller device e.g., a 10/100BaseT Octal MAC 13a or a Gigabit Ethernet device 13b. In general, as a network processor, the hardware-based multithreaded processor 12 can interface to any type of communication device or interface that receives/sends large amounts of data. Communication system 10 functioning in a networking application could receive network packets from the devices 13a, 13b and process those packets in a parallel manner. With the hardware-based multithreaded processor 12, each network packet can be independently processed.

Another example for use of processor 12 is a print engine for a postscript processor or as a processor for a storage subsystem, e.g., Redundant Array of Independent Disk (RAID) storage, a category of disk drives that employs two or more drives in combination for fault tolerance and performance. A further use is as a matching engine. In the securities industry for example, the advent of electronic trading requires the use of electronic matching engines to match orders between buyers and sellers. These and other parallel types of tasks can be accomplished utilizing the system 10.

The processor 12 includes the bus interface 28 that couples the processor to the second bus 18. In an embodiment, bus interface 28 couples the processor 12 to the FBUS (FIFO bus) 18. The FBUS interface 28 is responsible for controlling and interfacing the processor 12 to the FBUS 18. The FBUS 18 is a 64-bit wide FIFO bus, used to interface to Media Access Controller (MAC) devices, e.g., 10/100 Base T Octal MAC 13a.

The processor 12 includes a second interface e.g., PCI bus interface 24, that couples other system components that reside on the PCI 14 bus to the processor 12. The PCI bus interface 24 provides a high-speed data path 24a to memory 16, e.g., SDRAM memory 16a. Through PCI bus interface 24 data can be moved quickly from the SDRAM 16a through the PCI bus 14, via direct memory access (DMA) transfers. The hardware based multithreaded processor 12 supports image transfers. The hardware based multithreaded processor 12 can employ DMA channels so if one target of a DMA transfer is busy, another one of the DMA channels can take over the PCI bus 14 to deliver information to another target to maintain high processor 12 efficiency. Additionally, the

PCI bus interface 24 supports target and master operations. Target operations are operations where slave devices on bus 14 access SDRAMs through reads and writes that are serviced as a slave to a target operation. In master operations, the processor core 20 sends data directly to or receives data directly from the PCI interface 24.

Each of the functional units 22 is coupled to one or more internal buses. As described below, the internal buses are dual, 32 bit buses (i.e., one bus for read and one for write). The hardware-based multithreaded processor 12 also is constructed such that the sum of the bandwidths of the internal buses in the processor 12 exceed the bandwidth of external buses coupled to the processor 12. The processor 12 includes an internal core processor bus 32, e.g., an ASB Advanced System Bus (ASB), that couples the processor core 20 to the memory controller 26a, 26b and to an ASB translator 30, described below. The ASB bus 32 is a subset of the so-called Advanced Microcontroller Bus Architecture (AMBA) bus that is used with the Strong Arm processor core 20. AMBA is an open standard, on-chip bus specification that details a strategy for the interconnection and management of functional blocks that makes up a System-on-chip (SoC). The processor 12 also includes a private bus 34 that couples the micro engine units 22 to SRAM controller 26b, ASB translator 30 and FBUS interface 28. A memory bus 38 couples the memory controller 26a, 26b to the bus interfaces 24 and 28 and memory system 16 including flashrom 16c that is used for boot operations and so forth.

Referring to FIG. 2, each of the micro engines 22a-22f includes an arbiter that examines flags to determine the available threads to be operated upon. Any thread from any of the micro engines 22a-22f can access the SDRAM controller 26a, SDRAM controller 26b or FBUS interface 28. The memory controllers 26a and 26b each include queues to store outstanding memory reference requests. The queues either maintain order of memory references or arrange memory references to optimize memory bandwidth. For example, if a thread_0 has no dependencies or relationship to a thread_1, there is no reason that thread_1 and thread_0 cannot complete their memory references to the SRAM unit 16b out of order. The micro engines 22a-22f issue memory reference requests to the memory controllers 26a and 26b. The micro engines 22a-22f flood the memory subsystems 26a and 26b with enough memory reference operations such that the memory subsystems 26a and 26b become the bottleneck for processor 12 operation.

If the memory subsystem 16 is flooded with memory requests that are independent in nature, the processor 12 can perform memory reference sorting. Memory reference

sorting improves achievable memory bandwidth. Memory reference sorting, as described below, reduces dead time or a bubble that occurs with accesses to SRAM 16b. With memory references to SRAM 16b, switching current direction on signal lines between reads and writes produces a bubble or a dead time waiting for current to settle on conductors coupling the SRAM 16b to the SRAM controller 26b.

That is, the drivers that drive current on the bus need to settle out prior to changing states. Thus, repetitive cycles of a read followed by a write can degrade peak bandwidth. Memory reference sorting allows the processor 12 to organize references to memory such that long strings of reads can be followed by long strings of writes. This can be used to minimize dead time in the pipeline to effectively achieve closer to maximum available bandwidth. Reference sorting helps maintain parallel hardware context threads. On the SDRAM 16a, reference sorting allows hiding of pre-charges from one bank to another bank. Specifically, if the memory system 16b is organized into an odd bank and an even bank, while the processor is operating on the odd bank, the memory controller can start pre-charging the even bank. Pre-charging is possible if memory references alternate between odd and even banks. By ordering memory references to alternate accesses to opposite banks, the processor 12 improves SDRAM bandwidth. Additionally, other optimizations can be used. For example, merging optimizations where operations that can be merged, are merged prior to memory access, open page optimizations where by examining addresses an opened page of memory is not reopened, chaining, as will be described below, and refreshing mechanisms, can be employed.

The FBUS interface 28 supports Transmit and Receive flags for each port that a MAC device supports, along with an Interrupt flag indicating when service is warranted. The FBUS interface 28 also includes a controller 28a that performs header processing of incoming packets from the FBUS 18. The controller 28a extracts the packet headers and performs a micro programmable source/destination/protocol hashed lookup (used for address smoothing) in SRAM 16b. If the hash does not successfully resolve, the packet header is sent to the processor core 20 for additional processing. The FBUS interface 28 supports the following internal data transactions:

FBUS unit	(Shared bus SRAM)	to/from micro engine.
FBUS unit	(via private bus)	writes from SDRAM Unit.
FBUS unit	(via Mbus)	Reads to SDRAM.

The FBUS 18 is a standard industry bus and includes a data bus, e.g., 64 bits wide and sideband control for address and read/write control. The FBUS interface 28 provides the ability to input large amounts of data using a series of input and output FIFOs 29a-29b. From the FIFOs 29a-29b, the micro engines 22a-22f fetch data from or command the SDRAM controller 26a to move data from a receive FIFO in which data has come from a device on bus 18, into the FBUS interface 28. The data can be sent through memory controller 26a to SDRAM memory 16a, via a direct memory access. Similarly, the micro engines can move data from the SDRAM 26a to interface 28, out to FBUS 18, via the FBUS interface 28.

Data functions are distributed amongst the micro engines 22. Connectivity to the SRAM 26a, SDRAM 26b and FBUS 28 is via command requests. A command request can be a memory request or a FBUS request. For example, a command request can move data from a register located in a micro engine 22a to a shared resource, e.g., an SDRAM location, SRAM location, flash memory or some MAC address. The commands are sent out to each of the functional units and the shared resources. However, the shared resources do not need to maintain local buffering of the data. Rather, the shared resources access distributed data located inside of the micro engines 22a-22f. This enables micro engines 22a-22f, to have local access to data rather than arbitrating for access on a bus and risk contention for the bus. With this feature, there is a zero cycle stall for waiting for data internal to the micro engines 22a-22f.

The data buses, e.g., ASB bus 30, SRAM bus 34 and SDRAM bus 38 coupling these shared resources, e.g., memory controllers 26a and 26b, are of sufficient bandwidth such that there are no internal bottlenecks. In order to avoid bottlenecks, the processor 12 has a bandwidth requirement where each of the functional units is provided with at least twice the maximum bandwidth of the internal buses. As an example, the SDRAM 16a can run a 64 bit wide bus at 83 MHz. The SRAM data bus could have separate read and write buses, e.g., could be a read bus of 32 bits wide running at 166 MHz and a write bus of 32 bits wide at 166 MHz. That is, in essence, 64 bits running at 166 MHz that is effectively twice the bandwidth of the SDRAM.

The core processor 20 also can access the shared resources. The core processor 20 has a direct communication to the SDRAM controller 26a to the bus interface 24 and to SRAM controller 26b via bus 32. However, to access the micro engines 22a-22f and transfer registers located at any of the micro engines 22a-22f, the core processor 20 access the micro engines 22a-22f via the ASB Translator 30 over bus 34. The ASB translator 30 can physically reside in the FBUS interface 28, but logically is distinct. The ASB Translator 30 performs an address translation between FBUS micro engine transfer register locations and core processor addresses (i.e., ASB bus) so that the core processor 20 can access registers belonging to the micro engines 22a-22f.

Although micro engines 22a-22f can use the register set to exchange data as described below, a scratchpad memory 27 is also provided to permit micro engines 22a-22f to write data out to the memory for other micro engines to read. The scratchpad 27 is coupled to bus 34.

The processor core 20 includes a RISC core 50 implemented in a five stage pipeline performing a single cycle shift of one operand or two operands in a single cycle, provides multiplication support and 32 bit barrel shift support. This RISC core 50 is a standard Strong Arm® architecture but it is implemented with a five-stage pipeline for performance reasons. The processor core 20 also includes a 16-kilobyte instruction cache 52, an 8-kilobyte data cache 54 and a prefetch stream buffer 56. The core processor 20 performs arithmetic operations in parallel with memory writes and instruction fetches. The core processor 20 interfaces with other functional units via the ARM defined ASB bus. The ASB bus is a 32-bit bi-directional bus 32.

Referring to FIG. 3, an exemplary one of the micro engines 22a-22f, e.g., micro engine 22f, is shown. The micro engine 22f includes a control store 70, which, in one implementation, includes a RAM of here 1,024 words of 32 bit. The RAM stores a micro program (not shown). The micro program is loadable by the core processor 20. The micro engine 22f also includes controller logic 72. The controller logic 72 includes an instruction decoder 73 and program counter (PC) units 72a-72d. The four micro program counters 72a-72d are maintained in hardware. The micro engine 22f also includes context event switching logic 74. Context event logic 74 receives messages (e.g., SEQ_#_EVENT_RESPONSE; FBI_EVENT_RESPONSE; SRAM _EVENT_RESPONSE; SDRAM _EVENT_RESPONSE; and ASB _EVENT_RESPONSE) from each one of the shared resources, e.g., SRAM 26a, SDRAM

26b, or processor core 20, control and status registers, and so forth. These messages provide information on whether a requested function has completed. Based on whether or not a function requested by a thread has completed and signaled completion, the thread needs to wait for that completion signal, and if the thread is enabled to operate, then the thread is placed on an available thread list (not shown). The micro engine 22f can have a maximum of four threads available.

In addition to event signals that are local to an executing thread, the micro engines 22a-22f employ signaling states that are global. With signaling states, an executing thread can broadcast a signal state to all micro engines 22a-22f, e.g., Receive Request Available (RRA) signal, any and all threads in the micro engines 22a-22f can branch on these signaling states. These signaling states can be used to determine availability of a resource or whether a resource is due for servicing.

The context event logic 74 has arbitration for the four threads. In an embodiment, the arbitration is a round robin mechanism. Other techniques could be used including priority queuing or weighted fair queuing. The micro engine 22f also includes an execution box (EBOX) data path 76 that includes an arithmetic logic unit (ALU) 76a and general-purpose register set 76b. The ALU 76a performs arithmetic and logical functions as well as shift functions. The register set 76b has a relatively large number of general-purpose registers. In an embodiment, there are 64 general-purpose registers in a first bank, Bank A and 64 in a second bank, Bank B. The general-purpose registers are windowed so that they are relatively and absolutely addressable.

The micro engine 22f also includes a write transfer register stack 78 and a read transfer stack 80. These registers 78 and 80 are also windowed so that they are relatively and absolutely addressable. Write transfer register stack 78 is where write data to a resource is located. Similarly, read register stack 80 is for return data from a shared resource. Subsequent to or concurrent with data arrival, an event signal from the respective shared resource e.g., the SRAM controller 26a, SDRAM controller 26b or core processor 20 will be provided to context event arbiter 74, which will then alert the thread that the data is available or has been sent. Both transfer register banks 78 and 80 are connected to the execution box (EBOX) 76 through a data path. In an embodiment, the read transfer register has 64 registers and the write transfer register has 64 registers.

Referring to FIG. 4, the micro engine data path maintains a 5-stage micro-pipeline 82. This pipeline includes lookup of microinstruction words 82a, formation of the

register file addresses 82b, read of operands from register file 82c, ALU shift or compare operations 82d, and write-back of results to registers 82e. By providing a write-back data bypass into the ALU/shifter units, and by assuming the registers are implemented as a register file (rather than a RAM), the micro engine 22f can perform a simultaneous register file read and write, which completely hides the write operation.

The SDRAM interface 26a provides a signal back to the requesting micro engine on reads that indicates whether a parity error occurred on the read request. The micro engine micro code is responsible for checking the SDRAM 16a read Parity flag when the micro engine uses any return data. Upon checking the flag, if it was set, the act of branching on it clears it. The Parity flag is only sent when the SDRAM 16a is enabled for checking, and the SDRAM 16a is parity protected. The micro engines 22 and the PCI Unit 14 are the only requestors notified of parity errors. Therefore, if the processor core 20 or FIFO 18 requires parity protection, a micro engine assists in the request. The micro engines 22a-22f support conditional branches.

Referring to FIG. 5, a format for arithmetic logic unit instruction is shown. The micro engines 22 support various instruction sets. The instruction set includes logical and arithmetic operations that perform an ALU operation on one or two operands and deposit the result into the destination register, and update all ALU condition codes according to the result of the operation. Condition codes are lost during context swaps. When the op code bits 28:27 are 1:1 the instruction is a double shift instruction.

The instruction set includes a double shift instruction, i.e., DBL_SHF, which concatenates two long words (i.e., two 32 bit words) and shifts the result and saves the result as a longword. In the double shift instruction, the upper A-op shifts into lower B-op, with a "left rotate" of zero giving a zero shift (otherwise zero amount signifies indirect shift). The DBL_SHF instruction loads a destination register with a 32-bit longword that is formed by concatenating the A operands and B operands together, right shifting the 64-bit quantity by the specified amount, and storing the lower 32 bits.

A format of the double shift instruction is: *dbl_shf[dest_reg, A_operand, B_operand, A_op_shf_cntl],* where each of the fields is described fully below.

A "dest_req" field represents the destination, i.e., an absolute or context-relative register name.

A "A_operand" field represents a context-relative register name, i.e., 5-bit zero-filled immediate data.

A "B_oprand" field represents a context-relative register name, i.e., 5-bit zero-filled immediate data.

A "A_op_shf_cntl" field represents a right shift of values from 1 to 31.

By way of example, if a = 0x87654321 and b = 0xFEDCBA98, then dbl_shf[c, a, b, >>12] stores 0x321FEDCB in c. The ALU condition codes are updated based on the result.

It is to be understood that while the invention has been described in conjunction with the detailed description thereof, the foregoing description is intended to illustrate and not limit the scope of the invention, which is defined by the scope of the appended claims. Other aspects, advantages, and modifications are within the scope of the following claims.

What is claimed is:

1. A method of operating a processor comprising:
concatenating a first word and a second word to produce an intermediate result;
shifting the intermediate result by a specified shift amount; and
storing the shifted intermediate result in a third word.

2. The method of claim 1 wherein the first word and the second word and the third word are 32-bit words.

3. The method of claim 1 wherein the intermediate result is a 64-bit word.

4. The method of claim 1 wherein shifting comprises right shifting.

5. The method of claim 4 wherein the specified shift amount is in an operand.

6. The method of claim 4 wherein the specified shift amount is a value between one and thirty-one.

7. The method of claim 4 wherein the specified shift amount is a value contained in a lower five bits of the first word.

8. A computer instruction comprising:
an instruction to concatenate a first word and a second word to produce an intermediate result;
shift the intermediate result by a specified amount; and
store the shifted intermediate result in a third word.

9. The instruction of claim 8 wherein the first word and the second word and the third word are 32-bit words.

10. The instruction of claim 8 wherein the intermediate result is a 64-bit word.

11. The instruction of claim 8 wherein shifting comprises right shifting.

12. The instruction of claim 11 wherein the specified amount is in an operand.

13. The instruction of claim 11 wherein the specified amount is a value between one and thirty-one.

14. The instruction of claim 11 wherein the specified amount is a value contained in a lower five bits of the first word.

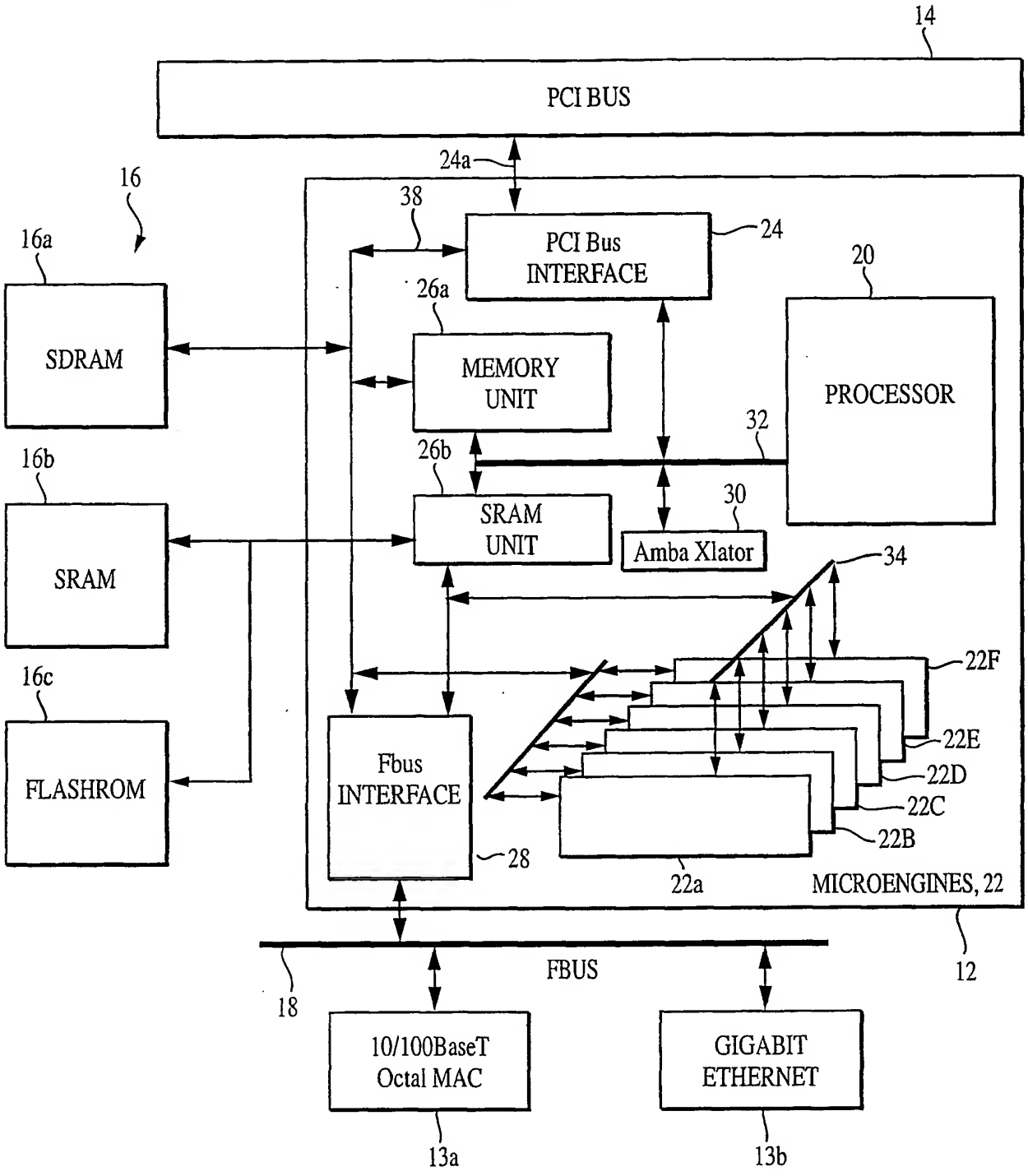

14
PCI BUS
24a
16
38
16a
PCI Bus INTERFACE
24
20
26a
SDRAM
MEMORY UNIT
PROCESSOR
32
16b
26b
30
SRAM
SRAM UNIT
Amba Xlator
34
22F
16c
22E
Fbus INTERFACE
22D
22C
FLASHROM
22B
28
22a
MICROENGINES, 22
12
FBUS
18
10/100BaseT Octal MAC
GIGABIT ETHERNET
13a
13b

FIG. 1

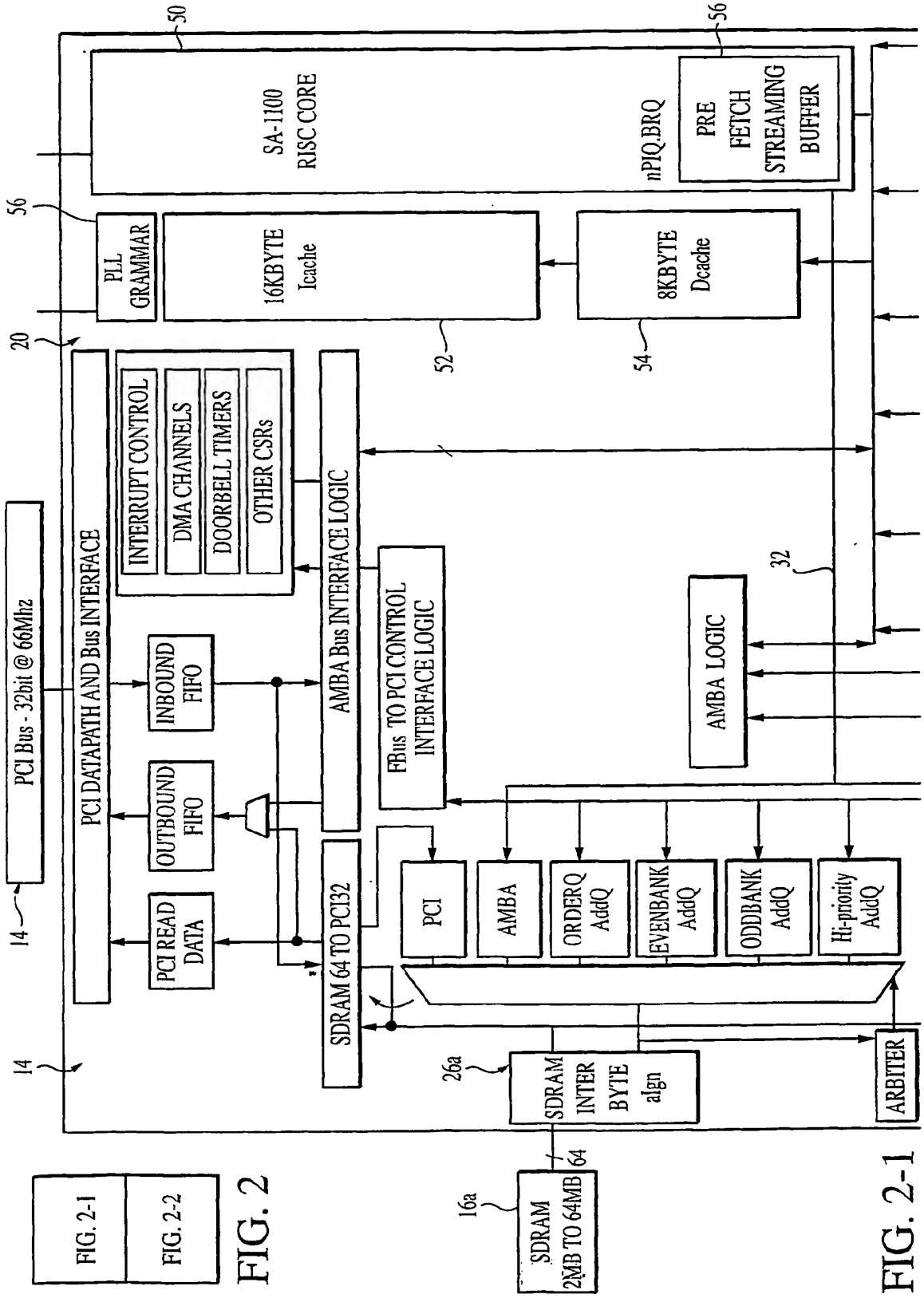


FIG. 2-1

FIG. 2-2

FIG. 2

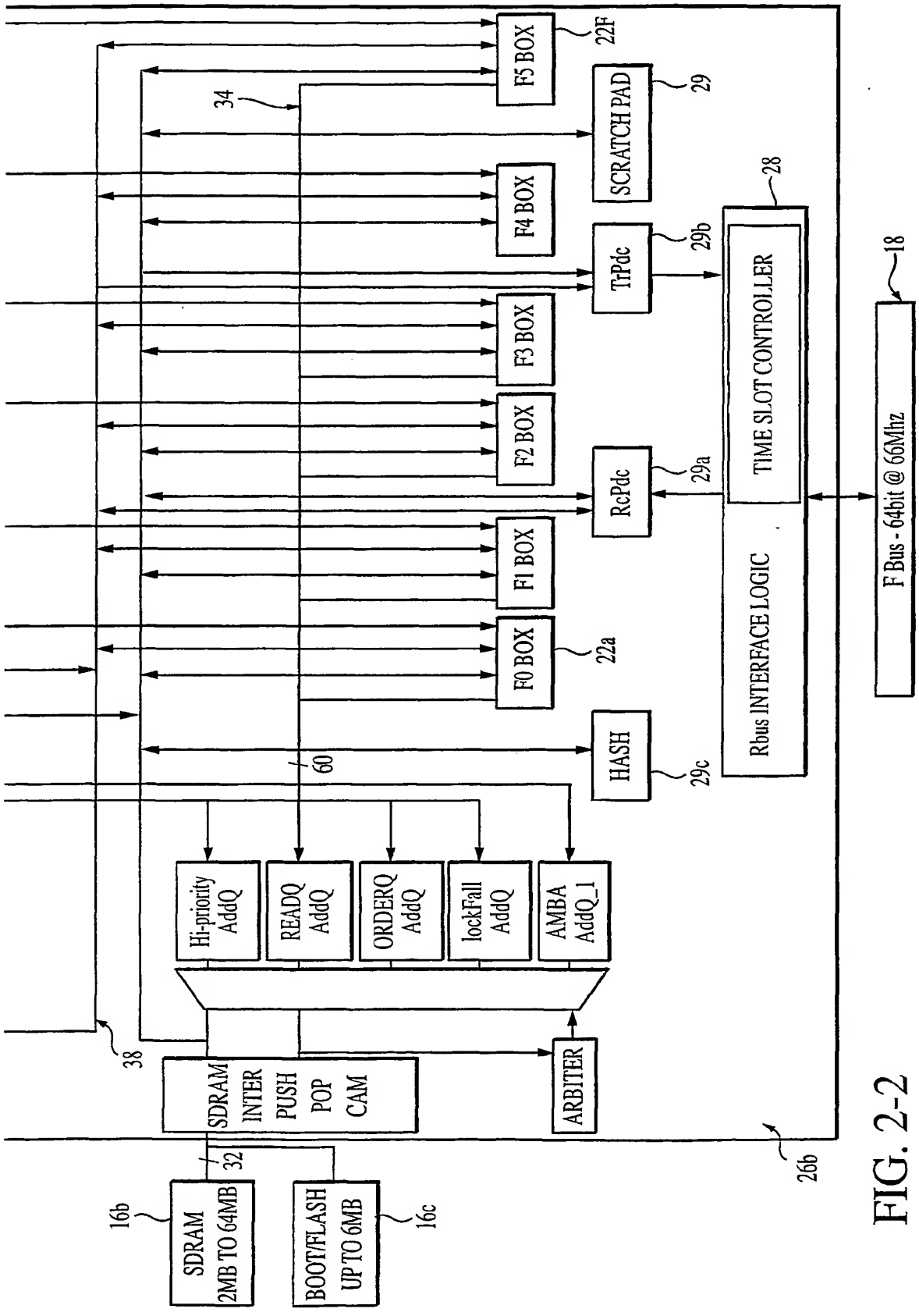

F5 BOX
22F
34
SCRATCH PAD
29
F4 BOX
28
TrPdc
29b
F3 BOX
TIME SLOT CONTROLLER
18
F2 BOX
RcPdc
29a
F1 BOX
F Bus - 64bit @ 66Mhz
F0 BOX
22a
Rbus INTERFACE LOGIC
HASH
29c
60
Hi-priority AddQ
READQ AddQ
ORDERQ AddQ
lockFail AddQ
AMBA AddQ_1
38
SDRAM INTER
PUSH
POP
CAM
ARBITER
26b
32
16b
SDRAM 2MB TO 64MB
BOOT/FLASH UP TO 6MB
16c

FIG. 2-2

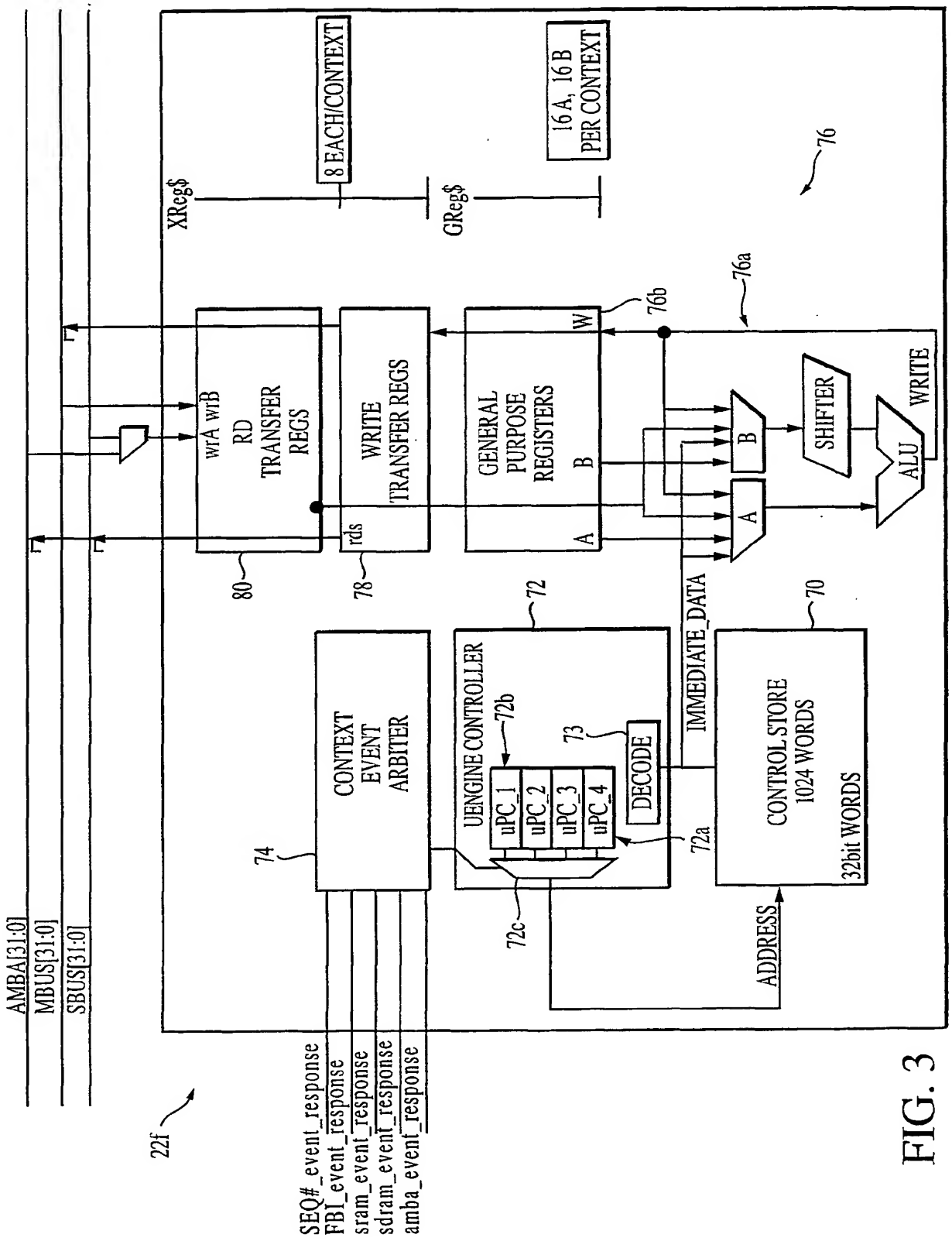

AMBA[31:0]
MBUS[31:0]
SBUS[31:0]
22f
SEQ#_event_response
FBI_event_response
sram_event_response
sdram_event_response
amba_event_response
XReg$
8 EACH/CONTEXT
GReg$
16 A, 16 B PER CONTEXT
76
76a
76b
wrA wrB RD TRANSFER REGS
WRITE TRANSFER REGS
GENERAL PURPOSE REGISTERS
W
B
A
rds
80
78
SHIFTER
ALU
WRITE
IMMEDIATE_DATA
72
70
CONTEXT EVENT ARBITER
UENGINE CONTROLLER
72b
73
uPC_1
uPC_2
uPC_3
uPC_4
DECODE
72a
72c
74
CONTROL STORE 1024 WORDS
32bit WORDS
ADDRESS

FIG. 3

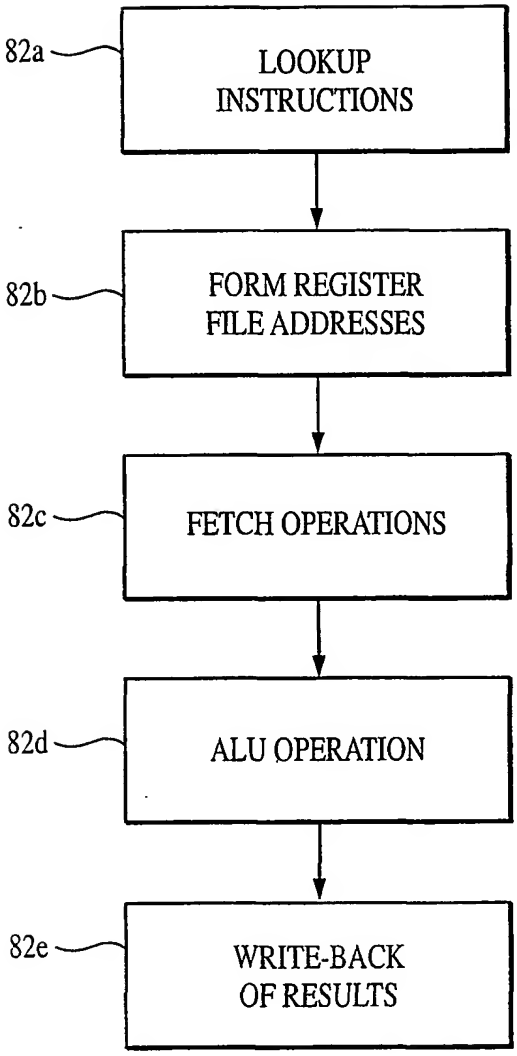
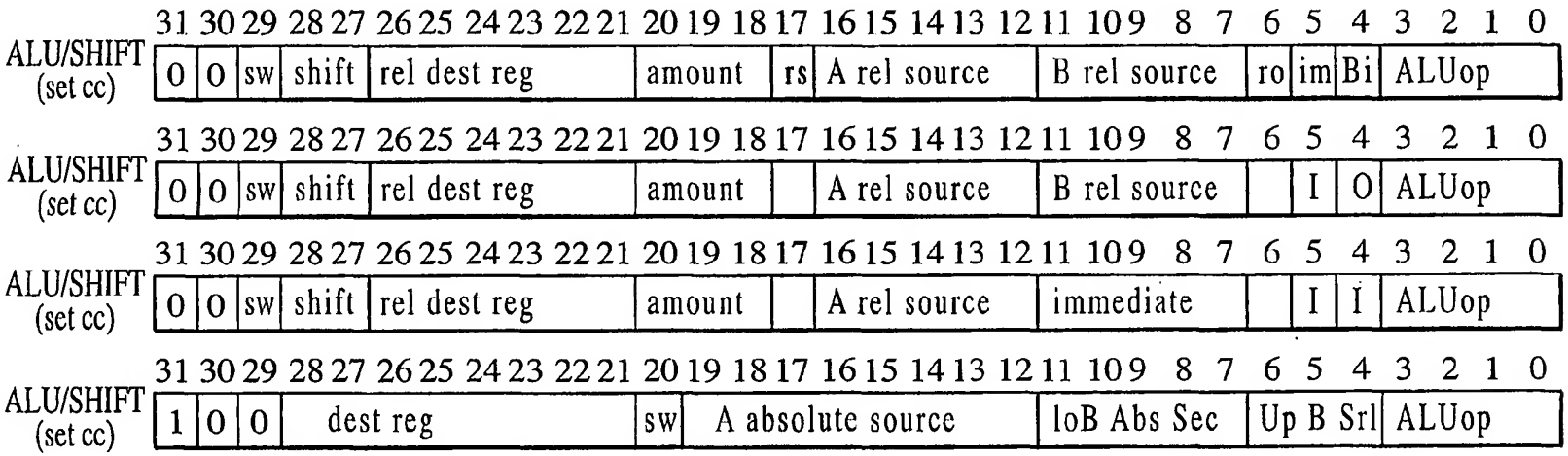

82a
LOOKUP INSTRUCTIONS
82b
FORM REGISTER FILE ADDRESSES
82c
FETCH OPERATIONS
82d
ALU OPERATION
82e
WRITE-BACK OF RESULTS

FIG. 4



Shift Decode:

(rs,r0) decode ([31:0] shifts into [63:32] and take [63;32]):

00 = left rotate
01 = right shift (32-ShfAmt = Right Shift Amt)
10 = left shift
11= double shift (upper A-op shifts into lower B-op)

===> "left rotate" of zero gives zero shift (therwise zero amount signifies indirect shift)

ALU-OP decode:

0000 = B	0100 = ~A&B (~and)	1000 = A-B	1100 = A+B(8)
0001 = ~B	0101 =XOR	1001 = B-A	1101 = A+B(16)
0010 = A&B (and)	0110 = OR	1010 =	1110 = A+B
0011= A&~B (and~)	0111= mul-stuff	1011=	0011= A+B+Cin

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/23982

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 9/315
US CL :712/204, 24; 711/220, 219

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 712/204, 24; 711/220, 219

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,717,760 A (SATTERFIELD et al.) 10 February 1998, note e.g. Figure 16.	1-14
Y	US 5,652,583 A (KANG) 29 July 1997, Figs 1 and 3, the Abstract and col.2.	1-14
Y	US 5,600,812 A (PARK) 04 February 1997, note the Abstract and cols. 3-5.	1-14
Y	US 5,436,626 A (FUJIWARA et al.) 25 July 1995, note the Abstract and Figure 2.	1-14
Y	US 5,363,448 A (KOOPMAN, Jr. et al.) 08 November 1994, note the Abstract and cols. 3-4.	1-14
A	US 5,113,516 A (JOHNSON) 12 May 1992, note the Abstract.	1-14

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

14 DECEMBER 2000

Date of mailing of the international search report

12 APR 2001

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

Peggy Harwood

B. JAMES PEIKARI

Telephone No. (703) 305-3824

Form PCT/ISA/210 (second sheet) (July 1998)★

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/23982

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 4,023,023 A (BOURREZ et al.) 10 May 1977, note the Abstract.	1-14

10/070,006

PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

(PCT Article 36 and Rule 70)

Applicant's or agent's file reference	FOR FURTHER ACTION	See Notification of Transmittal of International Preliminary Examination Report (Form PCT/IPEA/416)
10559-302WO1		
International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/US00/23982	31 AUGUST 2000	01 SEPTEMBER 1999
International Patent Classification (IPC) or national classification and IPC IPC(7): G06F 9/315 and US Cl.: 712/204, 24; 711/220, 219		
Applicant INTEL CORPORATION		

1. This international preliminary examination report has been prepared by this International Preliminary Examining Authority and is transmitted to the applicant according to Article 36.

2. This REPORT consists of a total of 4 sheets.

   ☐ This report is also accompanied by ANNEXES, i.e., sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications made before this Authority. (see Rule 70.16 and Section 607 of the Administrative Instructions under the PCT).

   These annexes consist of a total of — sheets.

3. This report contains indications relating to the following items:

   - I ☒ Basis of the report
   - II ☐ Priority
   - III ☐ Non-establishment of report with regard to novelty, inventive step or industrial applicability
   - IV ☐ Lack of unity of invention
   - V ☒ Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
   - VI ☐ Certain documents cited
   - VII ☐ Certain defects in the international application
   - VIII ☐ Certain observations on the international application

Date of submission of the demand	Date of completion of this report
08 MARCH 2001	01 DECEMBER 2002
Name and mailing address of the IPEA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer Peggy Harrod B. JAMES PEIKARI

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

International application No.
PCT/US00/23982

I. Basis of the report

1. With regard to the elements of the international application:*

[X] the international application as originally filed

[X] the description:
pages 1-11, as originally filed
pages NONE, filed with the demand
pages NONE, filed with the letter of _____

[X] the claims:
pages 12-13, as originally filed
pages NONE, as amended (together with any statement) under Article 19
pages NONE, filed with the demand
pages NONE, filed with the letter of _____

[X] the drawings:
pages 1-5, as originally filed
pages NONE, filed with the demand
pages NONE, filed with the letter of _____

[X] the sequence listing part of the description:
pages NONE, as originally filed
pages NONE, filed with the demand
pages NONE, filed with the letter of _____

2. With regard to the **language**, all the elements marked above were available or furnished to this Authority in the language in which the international application was filed, unless otherwise indicated under this item.
These elements were available or furnished to this Authority in the following language _____ which is:

[ ] the language of a translation furnished for the purposes of international search (under Rule 23.1(b)).

[ ] the language of publication of the international application (under Rule 48.3(b)).

[ ] the language of the translation furnished for the purposes of international preliminary examination (under Rules 55.2 and/or 55.3).

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international preliminary examination was carried out on the basis of the sequence listing:

[ ] contained in the international application in printed form.

[ ] filed together with the international application in computer readable form.

[ ] furnished subsequently to this Authority in written form.

[ ] furnished subsequently to this Authority in computer readable form.

[ ] The statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.

[ ] The statement that the information recorded in computer readable form is identical to the writen sequence listing has been furnished.

4. [X] The amendments have resulted in the cancellation of:

[X] the description, pages NONE

[X] the claims, Nos. NONE

[X] the drawings, sheets/~~fig~~ NONE

5. [ ] This report has been drawn as if (some of) the amendments had not been made, since they have been considered to go beyond the disclosure as filed, as indicated in the Supplemental Box (Rule 70.2(c)).**

** Replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report since they do not contain amendments (Rules 70.16*

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

International application No.
PCT/US00/23982

V. Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. statement

Novelty (N)	Claims	2-7 and 9-14	YES
	Claims	1 and 8	NO
Inventive Step (IS)	Claims	NONE	YES
	Claims	1-14	NO
Industrial Applicability (IA)	Claims	1-14	YES
	Claims	NONE	NO

2. citations and explanations (Rule 70.7)

Claims 1 and 8 lack novelty under PCT Article 33(2) as being anticipated by Satterfield et al. Note Figure 16 and the corresponding description of Figure 16, which explains how two values are added and then shifted to form a new word.

Claims 1 and 8 lack novelty under PCT Article 33(2) as being anticipated by Kang. Note Figures 1 and 3, the abstract and column 2, which explain how two words are added and then shifted to form a new word.

Claims 1 and 8 lack novelty under PCT Article 33(2) as being anticipated by Park. Note the abstract and columns 3-5, which explain how two words are added and then shifted two form a new word.

Claims 1 and 8 lack novelty under PCT Article 33(2) as being anticipated by Fujiwara et al.. Note the abstract and Figure 2, which explain how two codewords are added and then shifted to form a new word.

Claims 1 and 8 lack novelty under PCT Article 33(2) as being anticipated by Koopman, Jr et al. Note the abstract and columns 3-4, which explain how two numbers are added and then shifted to form a key word.

Claims 1-14 lack an inventive step under PCT Article 33(3) as being obvious over any one of the above-cited references (Satterfield et al., Kang, Park, Fujiwara et al., or Koopman, Jr. et al.) Each of these references fairly taught the subject matter of the claims, but not the particulars of the word length or direction of shifting etc. However, these particulars were well known and common knowledge to those working in this field. It would have been obvious, therefore, to incorporate these particulars into any one of the references listed above.

Claims 1-14 meet the criteria set out in PCT Article 33(4), because these claims find utility in such fields as parallel
(Continued on Supplemental Sheet.)

Supplemental Box
(To be used when the space in any of the preceding boxes is not sufficient)

Continuation of: Boxes I - VIII

V. 2. REASONED STATEMENTS - CITATIONS AND EXPLANATIONS (Continued):
processing, data encryption and address generation.

---------- NEW CITATIONS ----------
NONE